Amendments to the Specification:

Please amend page 1, lines 21 through lines 31 and page 2, lines 1 through lines 5 with the following rewritten lines:

In computing terms, a program is a specific set of ordered operations for a computer to perform. With an elementary form of programming language known as machine language, a programmer familiar with machine language can peek and poke data into and out from computer memory, and perform other simple mathematical transformations on data. Over time, however, the desired range of functionality of computer programs has increased quite significantly making programming in machine language generally cumbersome. As a result, proxy programming languages capable of being compiled into machine language and capable of much higher levels of logic have evolved. Examples of such evolving languages include COBOL®, Fortran®, Basic®, Pascal®, C®, C++®, Lisp®, Visual Basic®, C#® and many others. Some programming languages tend to be better than others at performing some types of tasks, but in general, the later in time the programming language was introduced, the more complex functionality that the programming language possesses empowering the developer more and more over time. Additionally, class libraries containing methods, classes and types for certain tasks are available so that, for example, a developer coding mathematical equations need not derive and implement the sine function from scratch, but need merely include and refer to the mathematical library containing the sine function.

Please amend page 3, lines 11 through lines 27 with the following rewritten lines:

Other source programs, such as dynamic link libraries (DLL) are collections of small programs, any of which can be called when needed by a larger program that is running in the computer. The small program that lets the larger program communicate with a specific device such as a printer or scanner is often packaged as a DLL program (usually referred to as a DLL file). DLL files that support specific device operation are known as device drivers. DLL files are an example of files that may be compiled at run-time.

The advantage of DLL files is that, because they don't get loaded into random access memory (RAM) together with the main program, space is saved in RAM. When and if a DLL file is needed, then it is loaded and executed. For example, as long as a user of Microsoft Word® is editing a document, the printer DLL file does not need to be loaded into RAM. If

the user decides to print the document, then the Word application causes the printer DLL file to be loaded into the execution space for execution.

A DLL file is often given a ".dll" file name suffix. DLL files are dynamically linked with the program that uses them during program execution rather than being compiled with the main program. The set of such files (or the DLL) is somewhat comparable to the library routines provided with programming languages such as Fortran®, Basic®, Pascal®, C®, C++®, C#®, etc.

Please amend page 6, lines 24 to lines 30 with the following rewritten lines:

C#® supports the above model, but in accordance with the present invention additionally supports an innovative language feature in this area, namely explicit interface member implementation. Explicit interface member implementation permits a class or struct to implement one or more interface members by explicitly specifying the relationship between the class or struct member and the interface member. This association is made by specifying the name of the class or struct member as a qualified name, consisting of the interface name and the interface member name.

Please amend page 12, lines 8 through lines 14 with the following rewritten lines:

Software may be designed using many different methods, including object-oriented programming methods. C++®, Java®, etc. are examples of common object-oriented programming languages that provide functionality associated with object-oriented programming. Object-oriented programming methods provide a means to encapsulate data members, e.g. variables, and member functions, e.g. methods, that operate on that data into single entity called a class. Object-oriented programming methods also provide means to create new classes based on existing classes.

Please amend page 12, line 21 through lines 31 and page 13, lines 1 through lines 31 with the following rewritten lines:

In exemplary embodiments of the explicit interface mechanism as described herein, the present invention is described in connection with the C#® programming language. However, one of ordinary skill in the art will readily recognize that the interfacing techniques of the present invention may be implemented with any programming language, such as Fortran®, Pascal®, Visual Basic®, C®, C++®, Java®, etc.

C#® is a simple, modern, object oriented, and type-safe programming language derived from C® and C++®. C#®, pronounced "C sharp" like the musical note, is firmly planted in the C® and C++® family tree of languages, and will be familiar to programmers having an understanding of the C® and C++® programming languages, and other object-oriented programming languages. Generally, C#® combines the high productivity of Visual Basic ®and the raw power of C++®, and provides many unique programming features as well.

C#® is provided as part of Microsoft Visual Studio® 7.0. In addition to C#®, Visual Studio® supports Visual Basic®, Visual C++®, and the scripting languages VBScript® and JScript®. All of these languages provide access to the Microsoft .NET® platform, which includes a common execution engine and a rich class library. The Microsoft .NET® platform defines a Common Language Subset (CLS), a sort of lingua franca that ensures seamless interoperability between CLS-compliant languages and class libraries. For C#® developers, this means that even though C#® is a new language, it has complete access to the same rich class libraries that are used by seasoned tools such as Visual Basic® and Visual C++®. C#® itself may not include a class library.

.Net® is a computing framework that has been developed in light of the convergence of personal computing and the Internet. Individuals and business users alike are provided with a seamlessly interoperable and Web-enabled interface for applications and computing devices, making computing activities increasingly Web browser or network-oriented. In general, the .Net® platform includes servers, building-block services, such as Web-based data storage and downloadable device software.

Generally speaking, the .Net® platform provides (1) the ability to make the entire range of computing devices work together and to have user information automatically updated and synchronized on all of them, (2) increased interactive capability for Web sites, enabled by greater use of XML (Extensible Markup Language) rather than HTML, (3) online

services that feature customized access and delivery of products and services to the user from a central starting point for the management of various applications, such as e-mail, for example, or software, such as Office .Net®, (4) centralized data storage, which will increase efficiency and ease of access to information, as well as synchronization of information among users and devices, (5) the ability to integrate various communications media, such as e-mail, faxes, and telephones, (6) for developers, the ability to create reusable modules, thereby increasing productivity and reducing the number of programming errors and (7) many other cross-platform integration features as well. While exemplary embodiments herein are described in connection with C#®, the interfacing of the present invention may be supported in all of Microsoft's .NET® languages. Thus, as one of ordinary skill in the art can appreciate, it would be desirable to incorporate the interfacing functionality of the present invention into any programming language.

Please amend page 14, lines 3 through lines 11 with the following rewritten lines:

The present invention provides an explicit interface member implementation in connection with a programming language, such as C#®. The explicit interface member implementation of the present invention permits a class or struct to implement one or more interface members by explicitly specifying the relationship between the class or struct member and the interface member. Explicit interface member implementation is a language feature that differentiates a programming language, such as C#®, from all other known computer programming platforms and languages. Any language that includes interface implementation could potentially benefit by adding a feature similar to the below-described explicit interface member implementation in C#®. Explicit interface member implementations

Please amend page 20, lines 25 through lines 31 and page 21, lines 1 through line 4 with the following rewritten lines:

As mentioned above, while exemplary embodiments of the present invention have been described in connection with the C#® programming language, the underlying concepts may be applied to any programming language for which it would be desirable to have

interfacing as described herein. Thus, an explicit interface member in accordance with the present invention may be implemented with any programming language, such as Fortran®, Pascal®, Visual Basic®, C®, C++®, Java® etc. In accordance with the present invention, a developer may utilize, or code, with a programming feature, namely an explicit interface member mechanism as described above, that enables a software component to implement an explicit interface member by explicitly specifying the relationship between the software component and the interface member.

Please amend page 21, lines 28 through lines 31 and page 22, lines 1 through lines 13 with the following rewritten lines:

While the present invention has been described in connection with the preferred embodiments of the various figures, it is to be understood that other similar embodiments may be used or modifications and additions may be made to the described embodiment for performing the same function of the present invention without deviating therefrom. For example, while exemplary embodiments of the invention are described in the context of programming in a networked or .NET® computing environment, one skilled in the art will recognize that the present invention is not limited thereto, and that the methods of programming in a programming environment having explicitly defined interface members, as described in the present application may apply to any computing device or environment, such as a gaming console, handheld computer, portable computer, etc., whether wired or wireless, and may be applied to any number of such computing devices connected via a communications network, and interacting across the network. Furthermore, it should be emphasized that a variety of computer platforms, including handheld device operating systems and other application specific operating systems are contemplated, especially as the number of wireless networked devices continues to proliferate. Therefore, the present invention should not be limited to any single embodiment, but rather should be construed in breadth and scope in accordance with the appended claims.